# Paving the way towards Abinit 8 in the post-Moore's law era

Yann Pouillon[1], Matteo Giantomassi[2], Jean-Michel Beuken[2], Thierry Deutsch[3], Damien Caliste[3], Xavier Gonze[2]

[1] *Euskal Herriko Unibertsitatea & ETSF Spain, Donostia-San Sebastián, Spain*

[2] *NAPS, Université catholique de Louvain, Louvain-la-Neuve, Belgium*

[3] *L_Sim, Commissariat à l'Énergie Atomique, Grenoble, France*

Dinard, France — 2013/04/15

# Summary of the previous episode

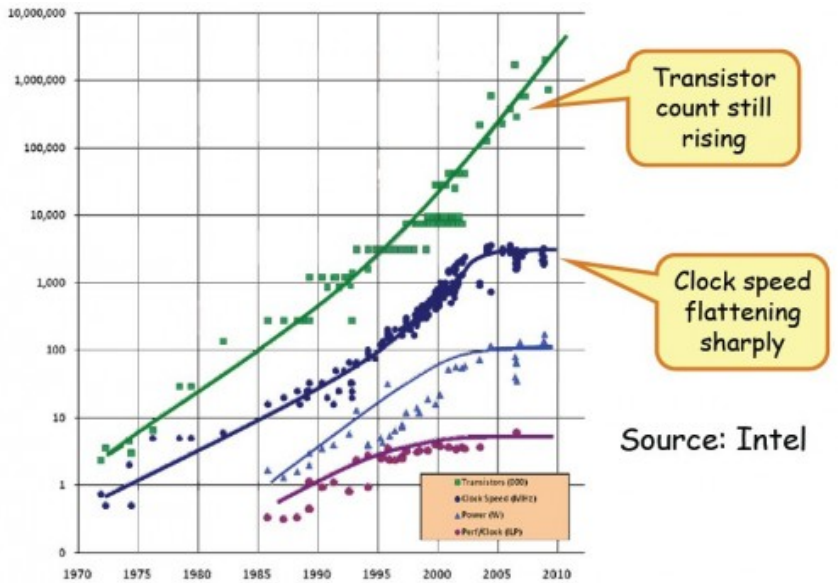
## Abinit 6 $\longrightarrow$ Abinit 7

*"Age is the acceptance of a term of years. But maturity is the glory of years."*

— Martha Graham

- Integrate 3 years of developments & expansion
- Mark a change in paradigms: serial $\rightarrow$ scaling
- Validate setting of test farm
- Validate phase separation of internal routines
- Validate first stage of block splitting (build systems)
- Give a new impulse

# A change in context




- 2012: core frequencies down
- 2020: serial code unusable
- Change paradigms now!
- Abinit: in progress (PRACE)

# New times, new ways

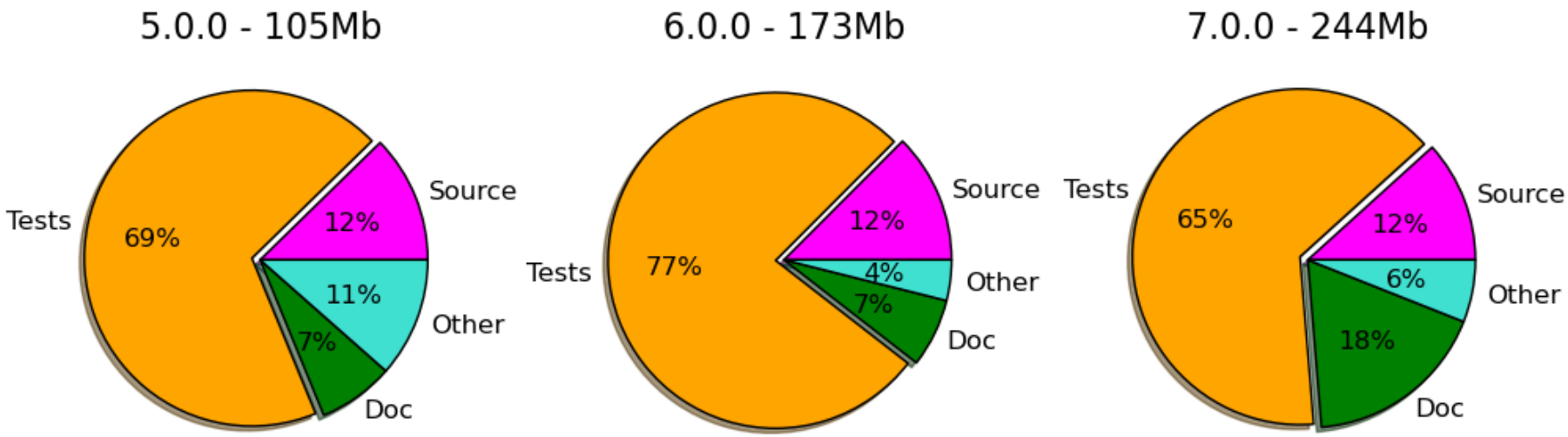
## Abinit 7 $\longrightarrow$ Abinit 8

*"You are never too old to become younger!"*

— Mae West

- End of Moore's law $\Longrightarrow$ scale, scale, scale
- Open data & high throughput $\Longrightarrow$ test, test, test
- Workflow-based methodology $\Longrightarrow$ split, split, split
- Rapidly evolving context $\Longrightarrow$ prune, prune, prune
- Nanotechnology $\Longrightarrow$ collaborate, collaborate, collaborate

# The weight of years




| Block | Contributors | Growth (Mb/year) | Rhythm (events/month) | Conflicts (%) |
|---|---|---|---|---|
| Source | 60 | 4 | 500 commits | 97 |
| Tests | 10 | 20 | 3 new tests | 0 |
| Doc | 10 | 8 | 3 updates | 0 |
| Other | 5 | Variable, $< 1$ | Variable | 3 |

# Skimming the Fatboy Abinit soup

**Ideal workflows**

| Block | Forge | Validation | Documentation |
|---|---|---|---|
| S+O = Core | Current structure | Unitary tests | Self-generated |
| Tests | One branch | Full runs | Self-generated |
| Doc | One branch | Builds | Included |

## Proposals (to be discussed)

1. Test Farm: split builds from tests
2. Web interface to add & edit full tests
3. Delegate build systems of test suite & documentation
4. Create team to design unitary tests
5. Create team to upgrade documentation

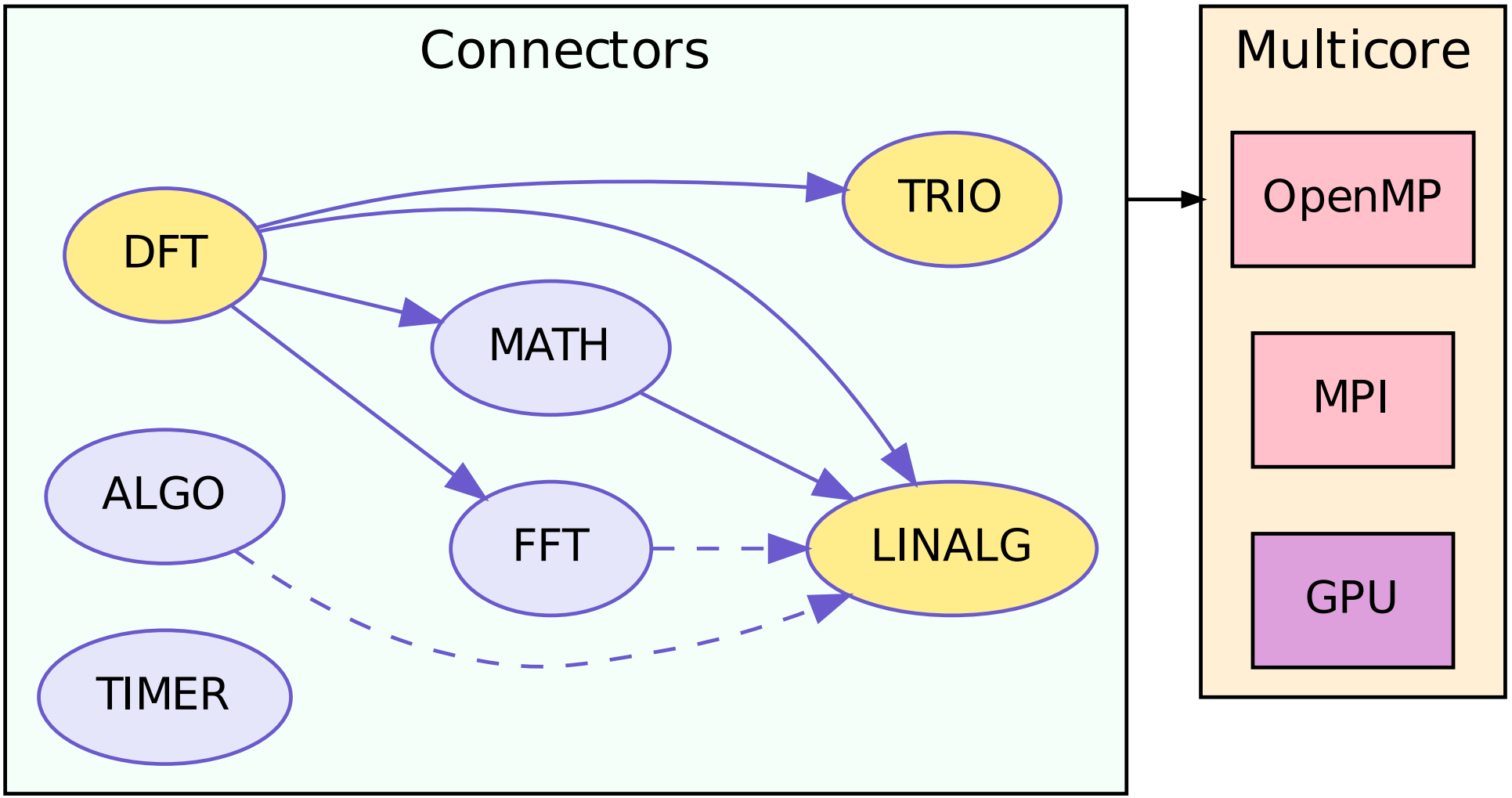
# Abinit 8: a rejuvenated project

- Core: 6 $\times$ smaller $\Longrightarrow$ Bazaar 6 $\times$ faster
- Tests: no download & easier maintenance if remote management
- Documentation: web-based management possible
- Test Farm: new workflows
- Fully independent build systems
  $\hookrightarrow$ Core: Yann Pouillon
  $\hookrightarrow$ Tests: Matteo Giantomassi?
  $\hookrightarrow$ Doc: Volunteer?

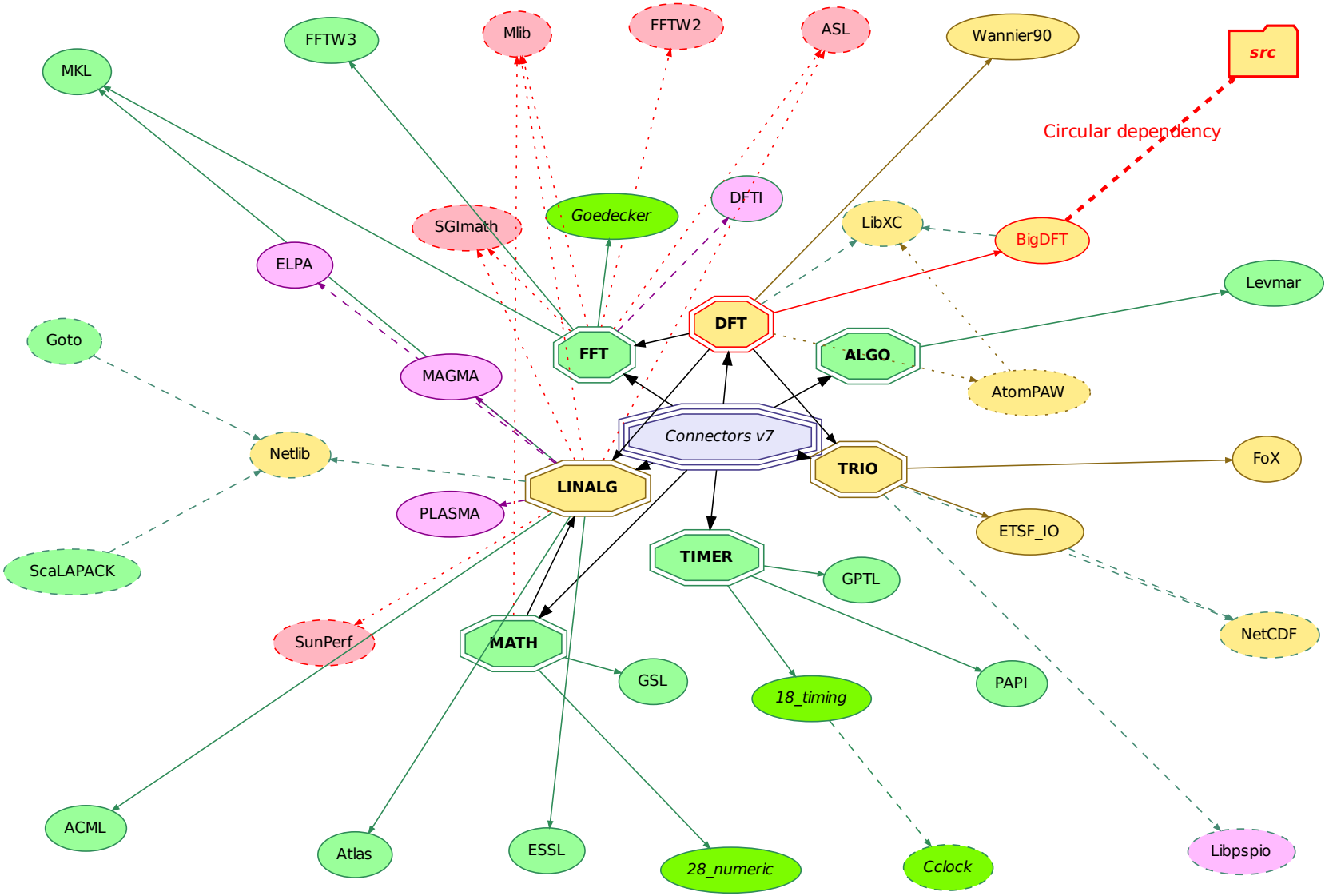
## Wait! There's some fat left there ...

- Split UIs from core: only one more branch
- Most dependencies unrelated to Abinit
- Create unique basis for common needs
- Easier to unify look & feel
- Easier to disentangle Abinit & UIs bugs

# Connecting external libraries



# Connectors — Abinit 7

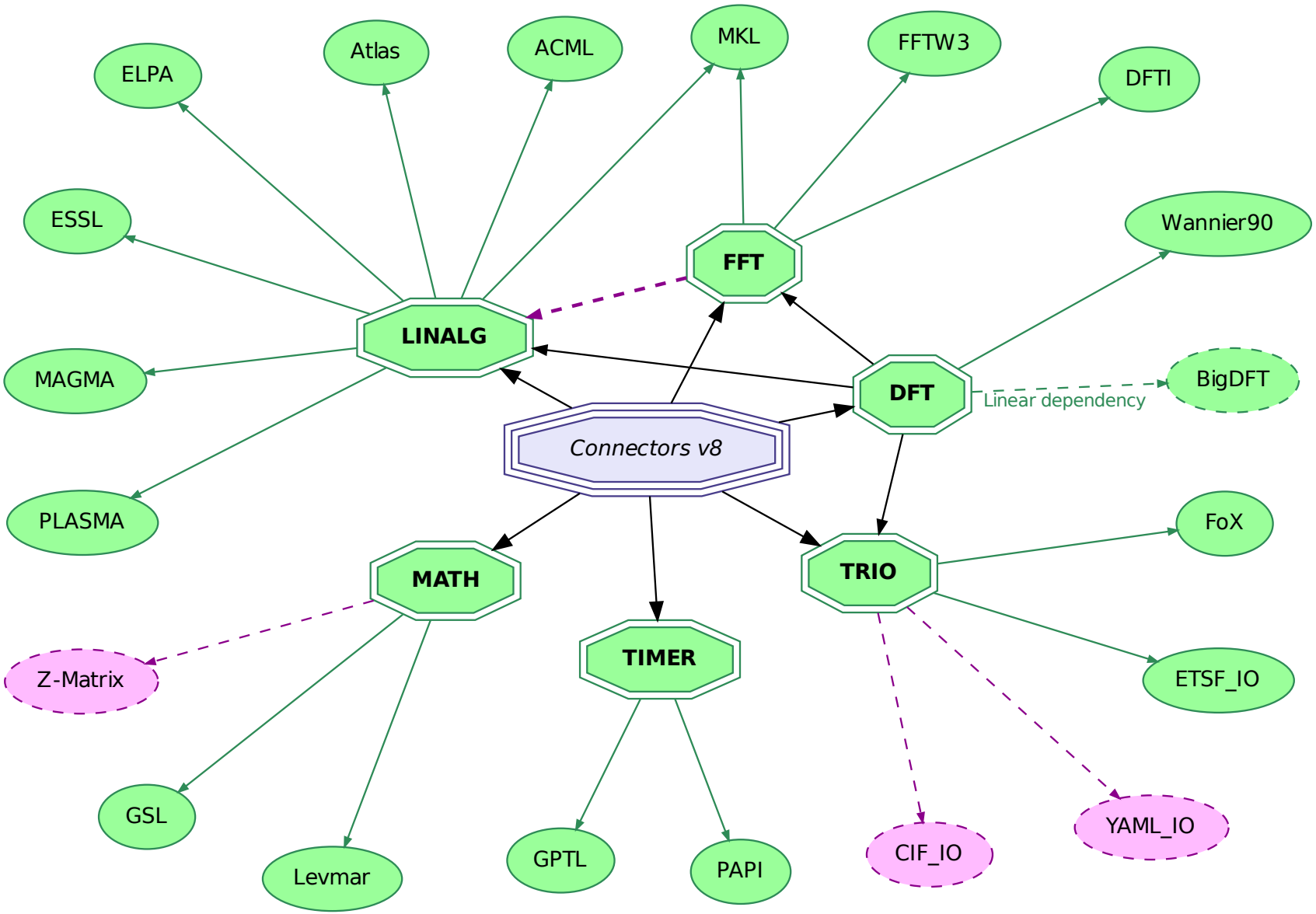


# Balancing complexity

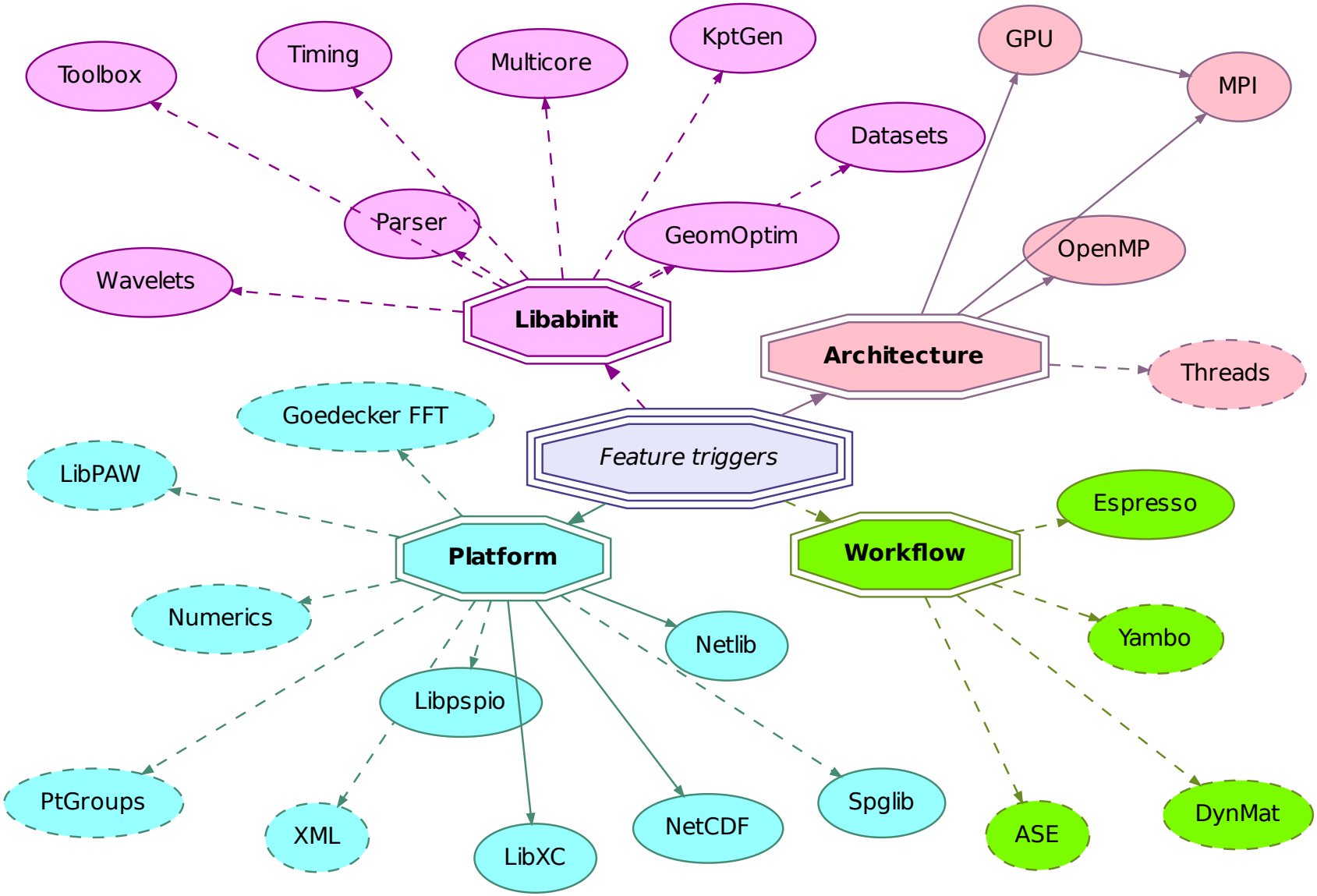
- **Forbid circular dependencies explicitely in abirules**
- Replace fallbacks by build instructions & external project
- Remove unused connector flavors
- Remove dependencies on obsolete libraries
- Remove AtomPAW fallback: no dependency
- Split necessary from optional libraries

- **New configure stage: feature triggers**
- Take decisions earlier
- Manage interactions: Architecture, Platform, Workflow, Abinit
- Refactor complex connectors (e.g. Netlib)
- Prepare externalization of basic components
- Included: LibXC, Libpspio, NetCDF, Netlib

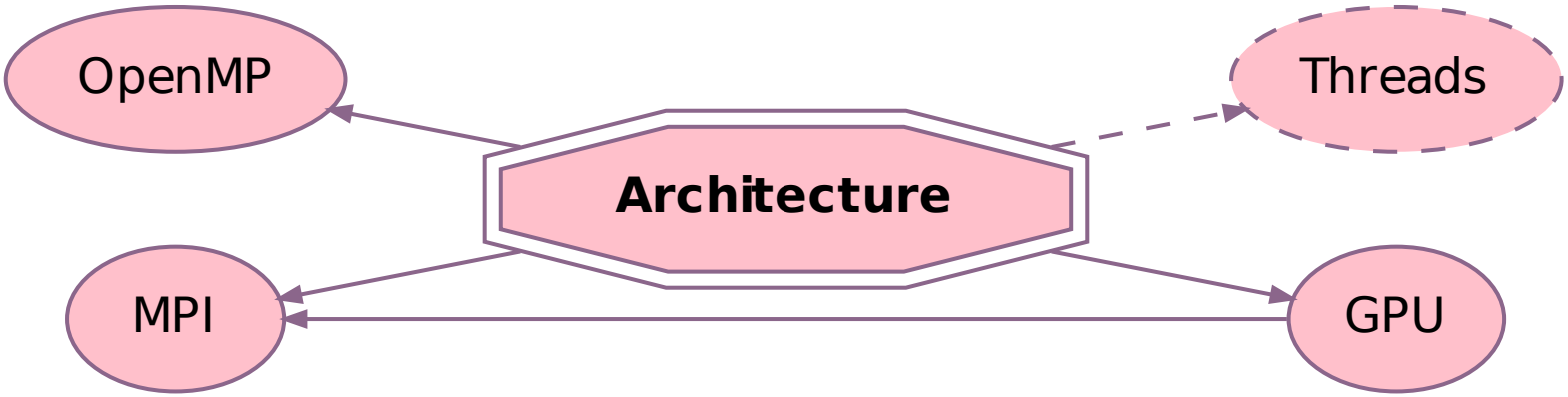
# Connectors — Abinit 8



# Feature triggers



# Adapting to complex architectures

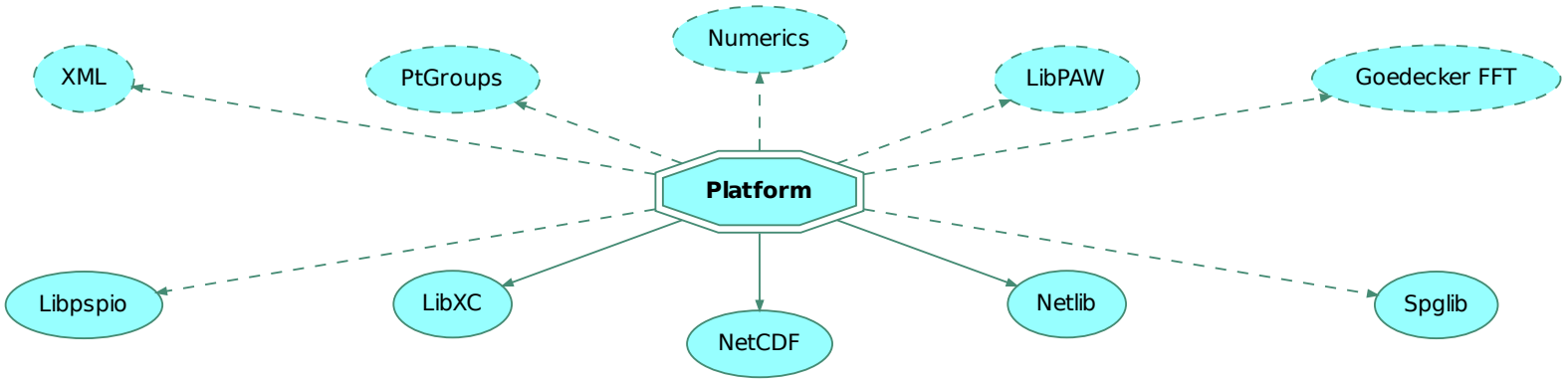



- Within a computing node: OpenMP, Threads
- Between computing nodes: MPI and/or GPU
- Adapt to heterogeneous environments
- Provide multi-scale parallelism
- Decision helper for Abinit
- See Matteo Giantomassi's talk on parallelism

# Allowing workflow-based runs

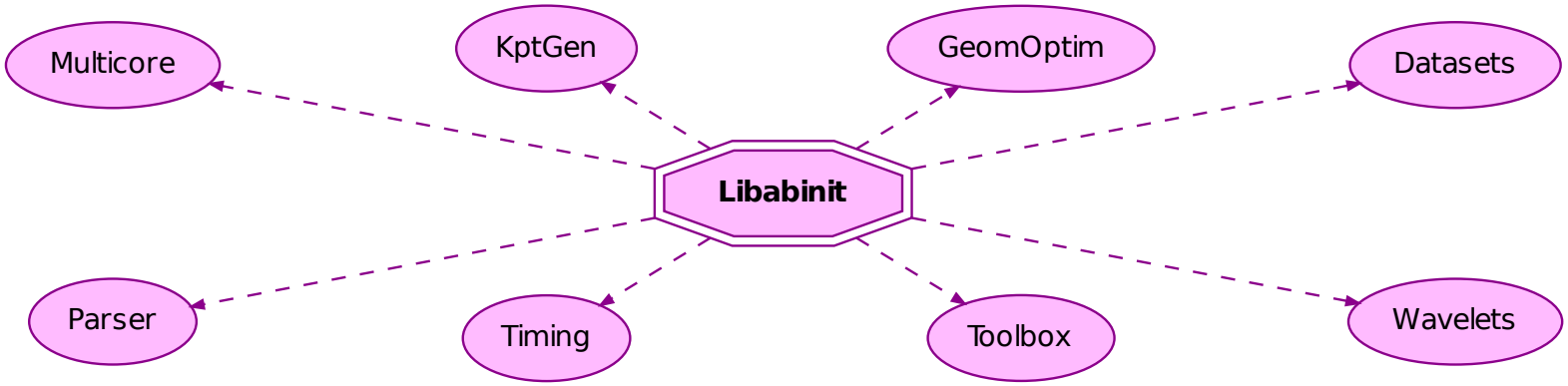

- Provide well-defined paths for data exchange
- Complex systems $\Longrightarrow$ multiple codes (*Quantum Espresso, Yambo, ...*)
- Complex calculations $\Longrightarrow$ integration (*ASE, GUIs, Test Farm, ...*)
- Increasing sizes $\Longrightarrow$ multi-scale modeling (*e.g. DFT + tight-binding*)
- Facilitate external contributions & reviews
- See Ask Hjorth Larsen's talk on ASE
- See session 7 on GUIs + Test Farm
- See Gabriel Antonius' talk on Abipy

# Taking benefit from the platform




- Explore common libraries providing basic features
- DFT/PW: I/O, linear algebra, FFT, maths, XC, pseudopotentials
- Use & reuse existing code & standards
- Contribute by externalizing Abinit components
- See Micael Oliveira's talk on Libpspio & LibXC

# The Abinit Library




- Common part: basic definitions, multicore, timing, toolbox
- DFT-related parts of interest to other codes
- Very strict naming conventions (e.g. *ab8_* prefix)
- Very clear API
- Very modular structure
- Fully informative
- *Discussion: complete list?*

# Bindings

- Up to now: Python bindings for the Abinit parser
  $\hookrightarrow$ *used by V_Sim*
- Increasing number of requests for more
- Step 1: disentangle BigDFT to free Abinit
- Step 2: add dynamic shared objects support to build system
- Step 3: release Libabinit

- *Discussion: solving the BigDFT Ouroboros*
- *Discussion: new bindings*

# Conclusion: the Abinit 8 todo list

1. Skim the Abinit Fatboy soup
2. Restructure the Forge & the Test Farm
3. Prune & refactor the source code
4. Develop the feature triggers
5. Refactor & optimize the connectors
6. Release Libabinit
7. Design & develop new bindings
8. Checkpoint: next Abinit Workshop

# Acknowledgments

***THANK YOU ALL!***